PIT-P2M : ProjectIT Process and Project Metamodel

Paula Ventura Martins [1], Alberto Rodrigues da Silva [2]

[1] INESC-ID, CSI/Universidade do Algarve
Campus de Gambelas, 8005-139 Faro, Portugal
pventura@ualg.pt
[2] INESC-ID/Instituto Superior Técnico
Rua Alves Redol, nº 9, 1000-029 Lisboa, Portugal
alberto.silva@acm.org

Abstract. Within the constant evolution observed in IT/IS area, new processes emerged faced to new customer's requirements and also due to new trends in software engineering community, such as unified or agile processes. Despite the great set of available tools in process and project management, still there is a real gap between "process" and "project" management approaches and respective tools. To assist team members on their work, the effort spend in a process customization could be used in other tasks, such as the project management task to control activities, work products and team members. In this paper, we describe a simplified SPEM-based metamodel for process specification and explain the motivation around this proposal. Considering this metamodel, we also propose a metamodel for project definition and configuration. To conclude, we demonstrate that this metamodel is better adapted to processes specification and can be applied in a project definition.

1 Introduction

The software engineering can be viewed according two basic dimensions: Process Management and Project Management. *Processes management* are pertinent techniques and tools applied to a process to implement and improve process effectiveness, hold the gains and ensure process integrity in fulfilling customer requirements [1].

Project Management is the application of knowledge, skills, tools and techniques to project activities to meet specific project requirements [2]. Project management is accomplished through the application and integration of project management tasks of initiating, planning, executing, monitoring and controlling and closing. The project goals must be followed in terms of costs, time and quality [3].

Process is an activities sequence with a set of inputs and outputs performed by specific roles during a time period. *Project* is a process instance that is performed according process guidelines. The adopted process in a project includes a set of predefined activities that can be planned and monitored in the scope of its own project management. Thus, it is evident an interconnection between process management and project management. The effort applied in a process configuration could be used in other tasks, such as project management tasks in order to control activities, products

In *Proceedings of the SEAA Conference*

and team members. The use of efficient processes can help in the success of project execution. However, it is necessary to define process characteristics before transiting for a project specification. In this paper, we propose a simplified SPEM-based process metamodel. Comparing our model with SPEM, we show that for an interaction between process and project specification, it is better to have a simple metamodel like that we propose.

This paper is organized in the following sections. Section 2 defines the process and project concepts. Section 3 describes related work, which somehow influence our work. Section 4 presents an overview of the ProjectIT initiative, whose main goal is to contribute with new ideas to improve the software development process. Section 5 describes the ProjectIT process metamodel, and presents its architecture and main functionalities, also presents the metamodel for projects definition. Finally, Section 6 summarizes general considerations on present possibilities and limits of the approach.

2 Process and Project Concepts

Process is a set of interrelated work activities characterized by a set of specific inputs and value added tasks that make up a procedure for a set of specific outputs. *Activities* are the units of work that sometimes may be related to each other, e.g., forming a hierarchy of activities, and they are associated to roles. *Roles* describe in an abstract form the set of skills and/or responsibilities associated with the execution of one or more activities. During activity enactment, developers create and transform work products. *WorkProduct* represent the object of work in an environment, and correspond to typical software development objects, such as requirements documents, test plans, test cases, etc. *Tools* automate execution of certain activities [4].

Processes are meant to be instantiated, resulting in an executable entity called a *project* (or simply process instance). The involving task of project coordination is called project management. Project management knowledge and practices are best described in terms of their component processes. These processes can be placed into five process groups (initiating, planning, executing, controlling and closing) and nine knowledge areas (project integration management, project scope management, project time management, project cost management, project quality management, project human resource management, project communications management, project risk management and project procurement management). *Project enactment* is guided by a project plan that might initially correspond to a parameterized instance of a generic process plan that is not necessarily complete. As work progresses, project plans may be adapted to project specific needs, or might be complemented with additional project specific definitions.

A *metamodel* defines a language for describing a specific domain of interest. For example UML (Unified Modelling Language) [5] is a language (i.e., a metamodel) to describe models for engineering systems. Some other metamodels address domains like process, organization, quality of service, etc. A process metamodel provides a set of generic concepts to describe any process models [6].

3 SPEM Metamodel

SPEM metamodel is used to describe a concrete process or a family of related processes [7]. The actual processes enactment, that is, project planning and executing, is not the scope of the SPEM metamodel. SPEM specification is presented as a UML profile and also provides a MOF based metamodel. This approach results from the large number of process models and standards, using each one a different terminology. SPEM allows different processes specifications, namely XP [8] [9], RUP [10] [11], MSF [12], DMR [7] [13], CMM [14] or ISO [15].

Although, UML is not necessarily tied to any application area or modelling process, its greatest applicability is in the area of object-oriented software design. UML is defined by a metamodel, which is itself defined as an instance of the MOF metametamodel. SPEM metamodel is described in a similar form as an UML subgroup extension called *SPEM_Foundation*. The SPEM metamodel is divided into four packages: *BasicElements*, *Depedencies*, *ProcessStructure*, *ProcessComponents* and *ProcessLifecycle*.

SPEM is a metamodel for process specification, so it is necessary to define its basic concepts in *ProcessStructure*, *ProcessComponents* and *ProcessLifecycle* packages. SPEM principles starts from the idea that a software development process is collaboration between abstract active entities called *Process Roles* that perform operations called *Activities* on concrete, tangible entities called *WorkProducts*.

3.1 SPEM Support Concepts

Fig. 1 defines the main structure elements (package *ProcessStructure*) that serve to construct a process description. In the diagram, the class *WorkProduct* is something that is produced, consumed or modified by a process, can be a document, a model or a source code. *WorkProductKind* describes a product category, could be a text document, a UML model, an executable, a code library or others. For that, each *WorkProduct* has associated a *WorkProductKind*. A *ProcessRole* is associated to each *WorkProduct*, which is formal responsible for its production. *WorkDefinition* is element of a process model that describes execution, operations and transformations performed in *WorkProducts* by the *ProcessRoles*. Its main subclass is *activity*, but *phase*, *iteration* and *lifecycle* are also its subclasses (Fig. 3). *ProcessPerformer* defines the responsible for a set of higher level *WorkDefinitions* in a process where individual roles cannot be associated. Its subclass *ProcessRole* defines responsibilities and abilities on specific *WorkProducts* and indicates the roles that perform or assist specific activities. Class *Activity* represents the work performed by a role, corresponding to tasks, operations and actions assisted or coordinated by a role. The atomic elements that constitute an *Activity* are called *Steps*.

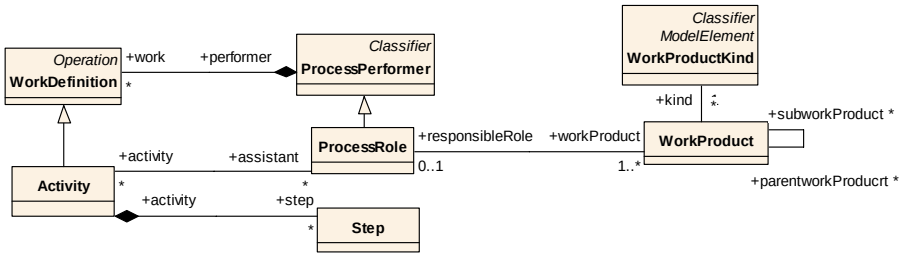

Fig. 1. Process structure (adapted from [7])

3.2 SPEM Process Components

Fig. 2 describes a process components package (sub-package *ProcessComponents*). Its classes are responsible for dividing one or more processes descriptions in "self-contained" parts that can be placed under configuration management or versions control. Such as in UML [5], a *Package* can both own and import process definition elements. A *ProcessComponent* is a process description that is internally consistent and can be reused with other process components to assemble a complete process. *Process* is a ProcessComponent intended to stand as a complete process from which it is possible to specify projects. It's distinguished from normal process components by the fact that it isn't intended to be composed with other components. *Discipline* is a particular specialization of *Package* that partitions the process activities according to a common "theme".

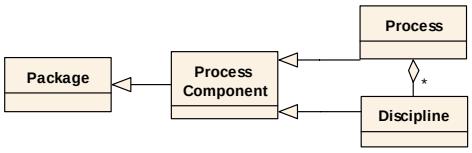

Fig. 2. Process components (adapted from [7])

3.3 SPEM Process Lifecycle

Package *LifeCycle* introduces process definition elements that help to represent its execution over time (Fig. 3). These elements describe or constrain the overall behaviour of the performing process and they are used to assist with planning, executing and monitoring the process. To coordinate its execution, activities order can be restricted. It is necessary to define the "shape" of the process over time and its lifecycle structure in terms of *Phases* and *Iterations*.

Phase is defined with the additional constraint of sequentially; that is, their enactments are executed with a series of milestone dates spread over time and often assume minimal (or no) overlap of their activities in time. A *Lifecycle* is associated with a sequence of *Phases*. An *Iteration* is a composed *WorkDefinition* but with a minor milestone (goal).

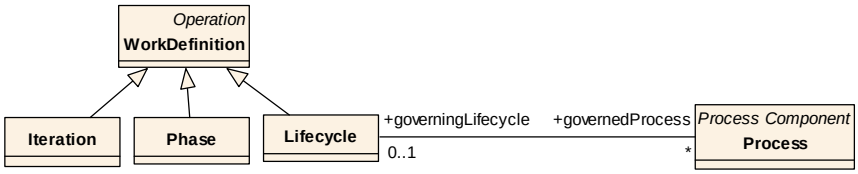

Fig. 3. Process lifecycle (adapted from [7])

4 ProjectIT Initiative

The INESC-ID Information Systems Group is a research group interested in topics related with software engineering and the software development process, and in applying them to the daily projects in which it is involved. ProjectIT is an R&D program that integrates some concrete issues related with information systems design, development and operation problems or, in general, the problematic of "projects in the area of information technologies"[16]. Its main goal is to provide a complete software development workbench, with support for project management, requirements engineering, analysis, design and code generation features.

ProjectIT intends to produce some results, namely (1) a collaborative tool with Web interface (i.e., with Web-client access) called “ProjectIT-Enterprise”; and (2) a rich-client tool (windows based) for improved productivity called “ProjectIT-Studio”. Both tools present tight complementarities and integration mechanisms. The Studio version has as its main goal to provide mechanisms for higher productivity to requirements management and specification, models design, automatic code generation and software development. On the other hand, the “ProjectIT-Enterprise” version provides mechanism to collaborative support for team work, emphasizing project management activities, workflows and documents management.

5 ProjectIT Metamodels

Considering the SPEM metamodel and comments like “*The actual enactment of processes—that is, planning and executing a project using a process described with SPEM, is not in the scope of this model*” and “*…minimal set of process modeling elements necessary to describe any software development process, without adding specific models or constraints for any specific area or discipline, such as project*

management or analysis" in an OMG document [7], make evidence specific and different needs in process and project models. SPEM address some concepts that are not essential in *project management* scope, these elements are: *WorkDefinition*, *Step*, *ProcessPerformer*, *Lifecycle* and *Guidance*. In project coordination there's no need to represent *WorkDefinition* as a composite pieces of work that are further decomposed in *Activities* and *Steps*. Class *ProcessPerformer* is not necessary considering his connection to *WorkDefinition*. *Lifecycle* is an abstract concept that describes how project time is organized in *Phases* and *Iterations*, without relevance in a project domain. SPEM package *BasicElements* has elements which contain a description of Model Elements, an essential feature in software process but without importance in project management discipline. These observations take us to a new approach considering new metamodels for project management based in SPEM.

PIT-ProcessM (ProjectIT Process Metamodel) is the component that supports, among other features, the definition of process models. On the other side, PIT-ProjectM (ProjectIT Project Metamodel) is a metamodel for projects definition. The main goal is to capture process concepts and mechanisms and apply them in concrete projects such as to monitories activities, evaluate products quality or people productivity. In this paper we present the PIT-ProcessM metamodel that allows the definition of process models and PIT-ProjectM metamodel for project creation.

5.1 Process Metamodel

Fig. 4 presents the PIT-ProcessM architecture, where is evident the relationships between the main process elements. The PIT-ProcessM defines the classes that correspond to elementary process concepts, allowing process creation or modification. Two complementary views show those static and dynamic process elements.

The static view describes *Activities*, *Roles*, *Workproducts* and *Disciplines*. Dynamic view describes how "things" append over time. An interface between this views is made by the class *Activity_Iteration*, where are specified the *Activities* that belongs to an *Iteration*.

Static View. An Activity represents the work performed by a role. But an activity can be decomposed in small work units, also called activities to unlimited deep of nested work. This concept is represented by the reflexive composed by aggregation. Control and data flow between activities is defined by the reflexive preceded by association. Activities produce and consume WorkProducts, which can also be formed by a set of small WorkProducts. Each WorkProduct is identified by a WorkProductKind, e. g., a document, a model, a source code, and so on. Activities are organized according to a common "theme" in Disciplines.

Dynamic View. The dynamic view identifies how process can be managed in terms of phases and iterations. Phases are defined with the additional constraint of sequentiality, with a series of milestones spread over time and often assume minimal overlap of their activities in time. Each phase include some iterations, which are activities flows but with small goals.

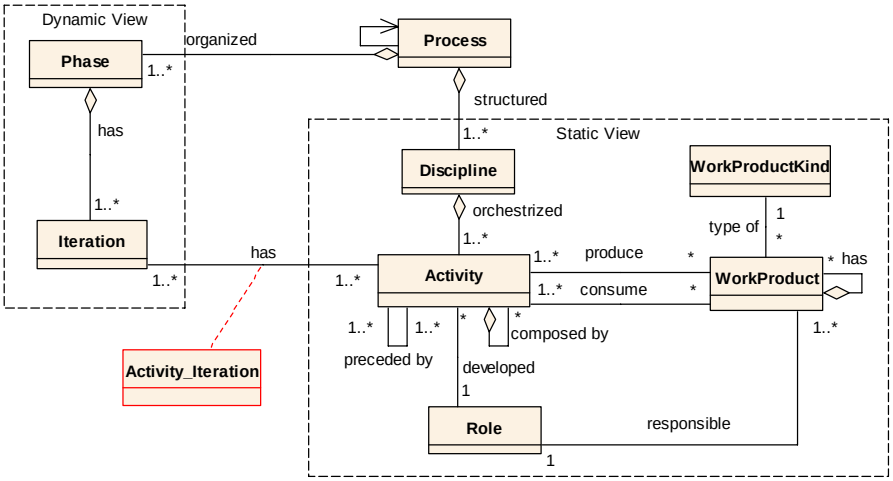

Fig. 4. ProjectIT Process Metamodel (PIT-ProcessM)

5.2 Project Metamodel

A project specification is based on the initial process definition model. *Projects* are frequently divided into more manageable components or subprojects, although the individual subprojects can be referred by themselves as projects and managed as such. *Projects* have associated specific information on *phases*, *iterations*, *activities* and *workproducts*. In a modelling perspective (Fig. 5) the differences between PIT-ProjectM and PIT-ProcessM metamodels are in the following classes: (1) *ActivityIterationProject*, which defines the activities to perform in each iteration; (2) *Person*, team members; (3) *PersonRoleProject* that defines the association between persons and project roles.

Associations between the classes *Person*, *WorkProductProject* and A*ctivityIterationProject* allow each person to visualize its responsibilities (activities and products), and verify relations to other persons work.

Team members can manage their time, place priorities in their activities, made decisions supported by data. Project manager has a global vision of project performance, being able to observe details in activities, iterations and team performance.

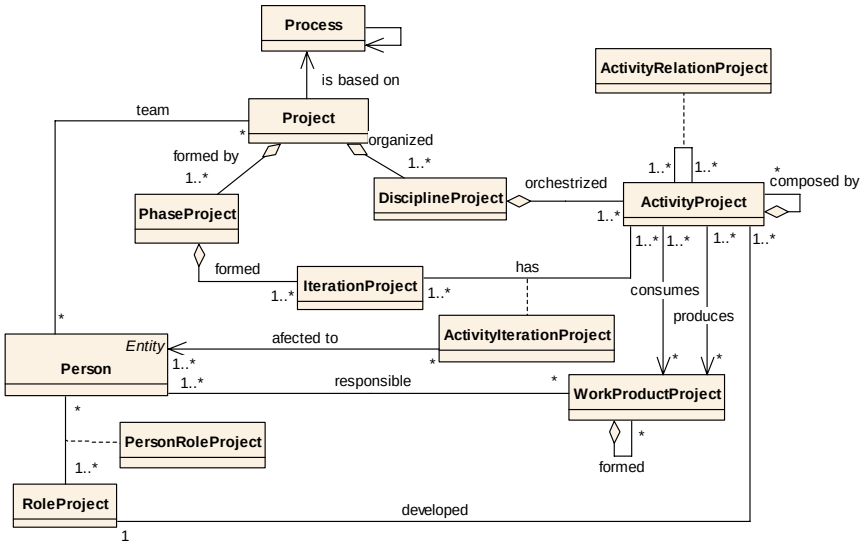

Fig. 5. ProjectIT Project Model (PIT-ProjectM)

6 Application Example with a XP Process

The following provides an application example of the proposed metamodel. For brevity, the example only covers the PIT-ProjectM metamodel for an agile project based on XP process. As seen in Fig. 5, the project model is based on the process defined according to PIT-ProcessM metamodel (Fig. 4).

The metamodel must be applied in all project Releases (*PhaseProject*) and Iterations. As an example we just considered the *fist Iteration* of *Release 1* and only one discipline (Planning). For this reason, classes *PhaseProject*, *IterationProject* and *DisciplineProject* are not instantiated in the UML diagram of Fig.6. The association that defines the *Person* responsible for a *WorkProductProject* is also omitted. Dependencies between project activities are not showed in this kind of diagram. Another UML activity diagram must be considered to represent relations between activities represented by class *ActivityRelationProject*.

The UML diagram of Fig. 6 represent *ActivityProject* performed by *Persons* (team members) and *WorkProductProject* produced or consumed by *ActivitiyProject*. Each *Person* only perform the kind of activities (*ActivityProject*) assigned to his *Role*. To understand the meaning of class symbols, a situation is described. The *Person* C1 (whose *Role* is a Customer) performs “Write Users Stories case A” and “Choose Next iteration User Stories” (*ActivityProject*). In the first activity the *WorkProductProject* produced are “User Stories from case A”. In the second activity, C1 takes his User Stories and selects the User Stories to be programmed in the next Iteration.

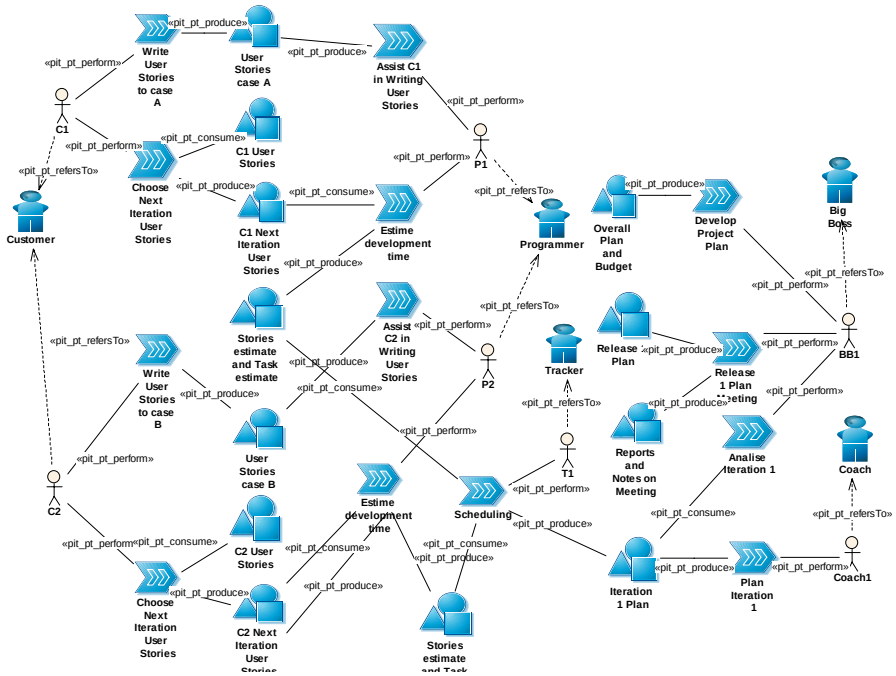

Fig. 6. Example of a XP project diagram based on PIT-ProjectM metamodel

7 Conclusions and Future Work

Table 1 describes relations between concepts of SPEM and PIT-ProcessM metamodels.

Table 1. Common concepts to SPEM and PIT-ProcessM metamodel

	Dynamic View			**Static View**			
SPEM	Phase	Iteration	Discipline	Activity	Step	Process Role	WorkProduct
PIT	Phase	Iteration	Discipline	Activity	Activity	Role	WorkProduct

SPEM metamodel is the standard for process specification, without specific models or constraints for any specific area or discipline, such as project management or analysis. Some SPEM basic elements are not important in a project management perspective. Elements like *WorkDefinition*, *Step*, *ProcessPerformer*, *Lifecycle* and *Guidance* are not relevant in planning, scheduling, collaboration and communication. The metamodel structure organized in four packages make clear that an

interconnection between their elements to support a project definition would result in a complex model. As an example, SPEM metamodels doesn't address a direct relationship between *disciplines* and *activities*. But in a project management perspective, it is essential to organize *activities* to make an easier project tracking.

If the metamodel is simplified, extension mechanisms could be adapted to specify project management issues integrated with metamodel elements. An important organizational feature to project management is team skills management, so it's essential to extend the models but keep them simple. Ours simplified SPEM based metamodels - PIT-ProcessM and PIT_ProjectM - can be applied in a process and project integrated environment to manage projects and teams. These UML models can be applied in a collaborative tool to support project management.

In this paper, we discussed some process and project concepts and their relationship to SPEM and PIT metamodels. We also demonstrate that a simplified process metamodel is better to define project features essential in project management.

References

1. Roger S. Pressman, "Software Engineering – A Practitioner´s Approach", McGraw Hill, 5th Edition, November 2003
2. David I. Cleland, William R. King, "Project Management Handbook", New York: Van Nostrand Reinhold, 1988.
3. Sommerville, "Software Engineering", Addison Wesley, 7th Edition, May 2004
4. P. Barthelmess,"Colaboration and coordination in process-centered software development environments: a review of the literature", I*nformation and Software Technology*, Elsevier, 2003, pp. 911-928.
5. OMG, "OMG Unified Modelling Language Specification version 1.3", OMG Document formal/02-11-14, November 2002.
6. C. Rolland, S. Nurcan, G. Grosz, "Enterprise knowledge development: the process view", Information & Management, vol. 36, pp. 165-184, 1999.
7. OMG, "Software Process Engineering Metamodel Specification", Version 1.1, January 2005, http://www.omg.org/technology/documents/formal/spem.htm (accessed in May of 2005).
8. K. Beck, "Extreme Programming Explained". Boston, MA: Addison-Wesley, 2000
9. R. Jeffries, A. Anderson, C. Hendrickson, "Extreme Programming Installed", Addison Wesley, 1st Edition, October 2000.
10. P. Kruchten, "The Rational Unified Process: An Introduction", Addison-Wesley Pub Co, 3rd Edition, December 2003.
11. Rational Unified Process (RUP), Rational Unified Process, Version 2003.06.01.
12. G. Lory, "Microsoft Solutions Framework", Version 3.0, http://www.microsoft.com/ technet/itsolutions/techguide/msf/msfovrvw.mspx (accessed in June of 2004).
13. DMR Consulting, "DMR Macroscope", Version 3.1, April 2000.
14. CMM, Capability Maturity Model for Software, http://www.sei.cmu.edu/cmm (accessed in June of 2004).
15. ISO, ISO/ICE 12207, http://www.software.org/quagmire/descriptions/isoiec12207 (accessed in June of 2004).
16. Alberto Rodrigues da Silva, "O programa de Investigação Project-IT", Technical report, V1.0, October 2004, INESC-ID, http://berlin.inesc.pt/alb/uploads/1/193/pit-white-paper-v1.0.pdf.